Service Classification through Machine Learning: Aiding in the Efficient Identification of Reusable Assets in Cloud Application Development

Zakieh Alizadehsani*, Daniel Feitosa†, Theodoros Maikantis‡, Apostolos Ampatzoglou‡, Alexander Chatzigeorgiou‡, David Berrocal*, Alfonso González Briones§, Juan M. Corchado*, Marcio Mateus¶ and Johannes Groenewold‖

*University of Salamanca, Spain
zakieh@usal.es, dabm@usal.es,corchado@usal.es
†University of Groningen, The Netherlands
d.feitosa@rug.nl
‡University of Macedonia, Greece
teomaik19@gmail.com, apostolos.ampatzoglou@gmail.com, achat@uom.edu.gr
§AIR Institute, Spain
alfonsogb@air-institute.org
¶Unparallel, Portugal
marcio.mateus@unparallel.pt
‖Contact Software, Germany
johannes.groenewold@contact-software.com

***Abstract*—Developing software based on services is one of the most emerging programming paradigms in software development. Service-based software development relies on the composition of services (i.e., pieces of code already built and deployed in the cloud) through orchestrated API calls. Black-box reuse can play a prominent role when using this programming paradigm, in the sense that identifying and reusing already existing/deployed services can save substantial development effort. According to the literature, identifying reusable assets (i.e., components, classes, or services) is more successful and efficient when the discovery process is domain-specific. To facilitate domain-specific service discovery, we propose a service classification approach that can categorize services to an application domain, given only the service description. To validate the accuracy of our classification approach, we have trained a machine-learning model on thousands of open-source services and tested it on 67 services developed within two companies employing service-based software development. The study results suggest that the classification algorithm can perform adequately in a test set that does not overlap with the training set; thus, being (with some confidence) transferable to other industrial cases. Additionally, we expand the body of knowledge on software categorization by highlighting sets of domains that consist *'grey-zones'* in service classification.**

***Index Terms*—web service, machine learning, service classification**

I. Introduction

With the advent of Open Source Software (OSS) and the continuous adoption of open practices, software reuse has become widely popular, due to the enormous amount of freely and openly available software assets (e.g., code, components, or services) [1]. Reuse of software assets is the process of using already available solutions to construct new software or enhance an existing one with new functionalities; thus, forgoing (or at least trying to minimize) the from-scratch development process [2]. Reuse is expected to bring important benefits to software development, especially with respect to *time to market* and the *quality of software* [3]. In the literature, there are two mainstream processes to reuse: systematic reuse, e.g., through product lines, model-driven engineering, etc. [4]; and opportunistic reuse, e.g., by searching development forums like StackOverflow for pieces of code [5], or OSS repositories for classes, libraries, or products [6]. As a first step towards reuse, the practitioner needs to perform ***reusable asset identification*** [7]. In this step, the reuser has to identify a piece of source code that implements the functionality that she/he wants to reuse. This is a very difficult task because: (a) the available amount of reusable assets in OSS is vast, and in some cases not well-organized and documented, and (b) there is a lack of platforms that can act as search engines for providing access to OSS repositories.

By considering the rise of the *'everything as a service (XaaS)'* model, as well as the current advancements in *cloud computing*, software development rapidly moves towards developing and deploying all software assets as services [8]. This emerging change has raised several challenges in software development [9], [10] and yields for specialized reuse processes. In SMARTCLIDE[1], an EU-funded research project, we have extended the Eclipse Theia IDE, which enables cloud application developers to perform reuse at three levels (ordered by priority): (a) ***in-house reuse***, by reusing services that have been developed and deployed internally; (b) ***domain-specific reuse***, by reusing services that are released as OSS, and the source and target systems belong to the same appli-

Work reported in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871177 (project: SMARTCLIDE).

[1]https://smartclide.eu/

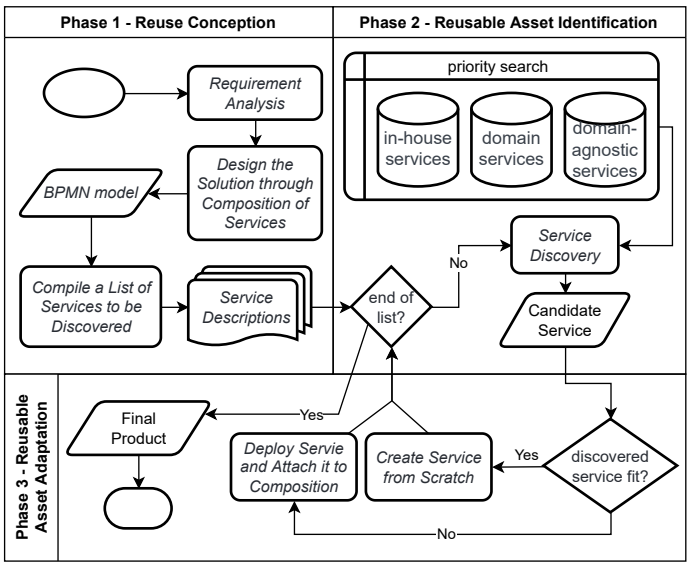

Fig. 1. SMARTCLIDE process for the reuse of software services

cation domain (e.g., games, business applications, etc.); and (c) ***domain-agnostic reuse***, by reusing OSS services from a different application domain. We clarify that ***domain-specific reuse*** is promoted in SMARTCLIDE, since software reuse is more efficient when performed within the same application domain [11].

To achieve this goal, in SMARTCLIDE, we have tailored the REACT reuse process [12] to fit the service-based software development paradigm, as illustrated in Fig. 1. In this paper, we focus on 'Phase 2' of the process. The component repository contains services that are labelled as either in-house or OSS. In-house components are developed from scratch by end-users, and are stored in SMARTCLIDE's repository. OSS components have been crawled from various OSS repositories (e.g., GitHub, ProgrammableWeb, etc.) and are pre-stored in the SMARTCLIDE's repository. So far, thousands of OSS services have been stored into SMARTCLIDE's repository. As a second level of labelling, the classification of OSS services to application domains needs to be performed. As it can easily be understood, the manual classification of services is not a viable solution, since it is time consuming, prone to errors, and does not allow the continuous automated population of the repository from in-house and OSS components.

To this end, in this paper, we employ a machine learning (ML) approach to classify the OSS services stored in SMARTCLIDE's repository. To explore the accuracy of this classification, we have performed a case study on SMARTCLIDE's two pilot cases. In particular, using the ML model pre-trained on OSS services, we classify 67 closed-source services of the two industrial partners (being completely agnostic to the training of the ML model) and contrast the automated classification to the expert judgement of the engineers of the two companies. Apart from the contribution of the software solution being able to classify services (given their name and description), through this empirical assessment, we expand the body of knowledge on domain-specific reuse by identifying application domains that are closely related conceptually and *"confuse"* the ML model.

II. Related Work

In this section, we present work related to our study, i.e., other attempts on classifying software services, based on the application domain that they refer to.

Kamath et al. [13] assert that classifying services based on textual description, input and output parameters is not sufficient. To this end, they propose an approach for web service classification that uses word embedding and ranks by semantic relatedness to a given query. Furthermore they present and compare results of other different classification techniques including Multilayer Perceptron (MLP), Naïve Bayes and more. In a later work, the same group proposed a data science-based approach for the classification of web services, based on morphological analysis and machine learning [14]. Their methodology includes three main stages, the extraction of the service's functional semantics using NLP (Natural Language Processing) techniques, the modeling of the service's features and finally the classification of the service. The evaluation was done using the descriptions of 1076 services from 9 different domains, producing an accuracy of 93%. Finally, it was concluded that the use of Multinomial Näive Bayes and Support Vector Machines (SVM) performed more poorly compared to the use of Ensemble methods for the proposed approach.

Crasso et al. [15] suggest that the difficulty of classifying a web service description, affects the adoption of Web Services negatively. They provided an automated approach (AWSC—Automatic Web Service Classification) to ease the process of classification by exploiting services described using standard Web Services Description Language (WSDL). The method used combines text mining and machine learning techniques in order to classify web services. The average accuracy that was achieved using the proposed methodology was 85%. Furthermore it was observed that AWSC accuracy results were superior by at least 9% when compared to related work based on Naıve Bayes and SVM.

Finally, Alshafaey et al. [16] proposed a new classification method called Cloud-Based Classification Methodology (CBCM). Similarly to Kamath and Ananthanarayana, CBCM consists of three main modules, (1) the Concepts Preparation Module (CPM) that scans and filters the inputted services in order to detect important keywords which are then conceptualized, (2) the Tree Creation Module (TCM) that creates a tree with the resulting concepts from module one and weighs them and finally (3) the classification module called CEAM (Change, Edit, Add Module) calculates and returns the best match based on the user's request. The dataset use for the validation consisted of 1007 services spread over 4 domains. The results showed that CBCM performed better in both precision and accuracy than other popular classification techniques such as SVM, etc.

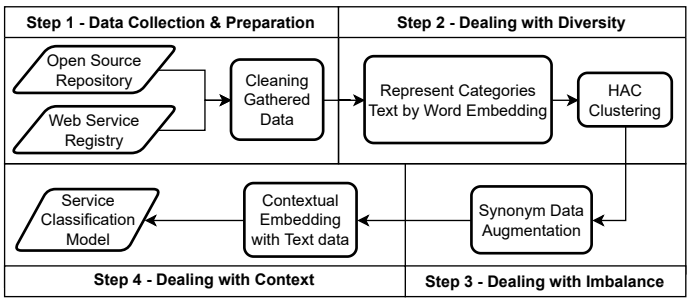

Fig. 2. Hybrid unsupervised / supervised model using contextual embedding

III. Proposed Classification Approach

Over the past decades, several approaches have been adopted to determine the category of a web service out of predefined categories [15], [17]–[23]. The early approaches have used manually defined services and keyword-based service discovery. Afterwards, semantics-based approaches were introduced to overcome keyword search limitations [17], [19]. In recent years, studies have focused on using AI-based techniques [18], [20], [23]. Although most earlier approaches have used information extraction for extracting service features from WSDL [15], [21], [22], [24], REST has become the prevalent solution for providing web services and APIs [25]. In RESTful service implementations, service description text data has become a significant feature in service classification. Accordingly, most text classification approaches have been applied in this research area. The earlier approaches mostly combined Bag of Words (BoW) and traditional ML. However, the keyword-based approaches such as BoW mostly use one-hot encoded vectors, which are high-dimensional and sparse.

The emergence of word-embedding techniques has improved keyword-based feature engineering. Additionally, the increasing word embedding of open-source projects such as Glove [24] and word2vec [26] helps the fast and efficient low-dimensional representation of web service data [17], [18], [23]. However, static word-embedding (e.g., word2vec, Glove) cannot capture the sequential information present in the text. To this end, recent studies have utilized contextual word-embedding (e.g., BERT) to provide a fast and efficient low-dimensional representation of web service metadata [23]. These approaches use modelling based on Bi-directional Long Short-Term Memory (BiLSTM) networks, offering better predictions than regular models. However, BiLSTM models suffer from high computational processes. Although Deep Learning (DL) algorithms have notable advantages, they also have some complexities, such as large data requirements, and high training/prediction time, which should also be considered.

In this study, we used a hybrid unsupervised and supervised model for service classification (see Fig. 2). The proposed model includes the following steps:

1. Collecting and preparing web service data from public resources: The current studies have shown that ProgrammableWeb is the most popular resource for training service classification models. This public service registry includes thousands of services from popular providers, such as Google APIs. Moreover, some additional data has been gathered from GitHub to prevent the lack of adequate coverage and deal with dataset skew.

2. Dealing with class diversity using service category clustering: The collected ProgrammableWeb dataset has a high degree of class diversity, negatively impacting classification accuracy. Therefore, the current work uses the top 50 categories of ProgrammableWeb and clustering algorithms to address this issue. To cluster service data, K-Means and Hierarchical Agglomerative Clustering (HAC), which are two popular clustering algorithms, have been investigated. K-Means attempts to partition the dataset into K predefined groups, which do not overlap. HAC does not need a predefined number of groups, as it uses a recursive concept, which can be applied bottom-up or top-down. Both the K-Means and HAC have been implemented. Ultimately, HAC showed better results and was selected.

Fig. 3 demonstrates which top 50 categories of ProgrammableWeb have high data skew. The left plot shows the 50 categories of ProgrammableWeb's dataset after dealing with imbalanced data. The right plot depicts the effect of data manipulation using data from GitHub, HAC, and synonym data augmentation. The data manipulation, led to the merging of several categories (e.g., 'Travel' with 'Transportation', and 'Advertising' with 'Marketing'), decreasing the number of possible classes from 50 to 42. The list of the original 50 classes as well as the final classes (with merged categories) are discussed in Section 5.2.

3. Dealing with imbalanced data using synonym data augmentation: The possible solutions for learning from imbalanced data include data-level and algorithm-level methods.

4. Providing a trained model for classification using contextual word-embedding: Pre-training methods such as BERT [27] have shown strong performance gains using self-supervised training that masks individual words or sub-word units for the formulated problem: $X = f(d, c)$, where d stands for service metadata, and c for service category text. The function f needs to provide relationships between service categories and web service metadata. To this end, the pre-training language model (BERT) has been applied to web service data. BERT uses mask word prediction to train a model in a special domain. This means that the model must be fine-tuned with service classification tasks. The fine-tuning has been applied in the benchmarking dataset of data gathered from ProgrammableWeb and GitHub. In this process, the original BERT's output layer is replaced with a new task-specific layer, which in this study is a service classification task. The results achieved with the training and evaluation set are shown in Fig. 4.

IV. Study Design

In this section, we present the results of the performed empirical study, in order to evaluate the accuracy of the

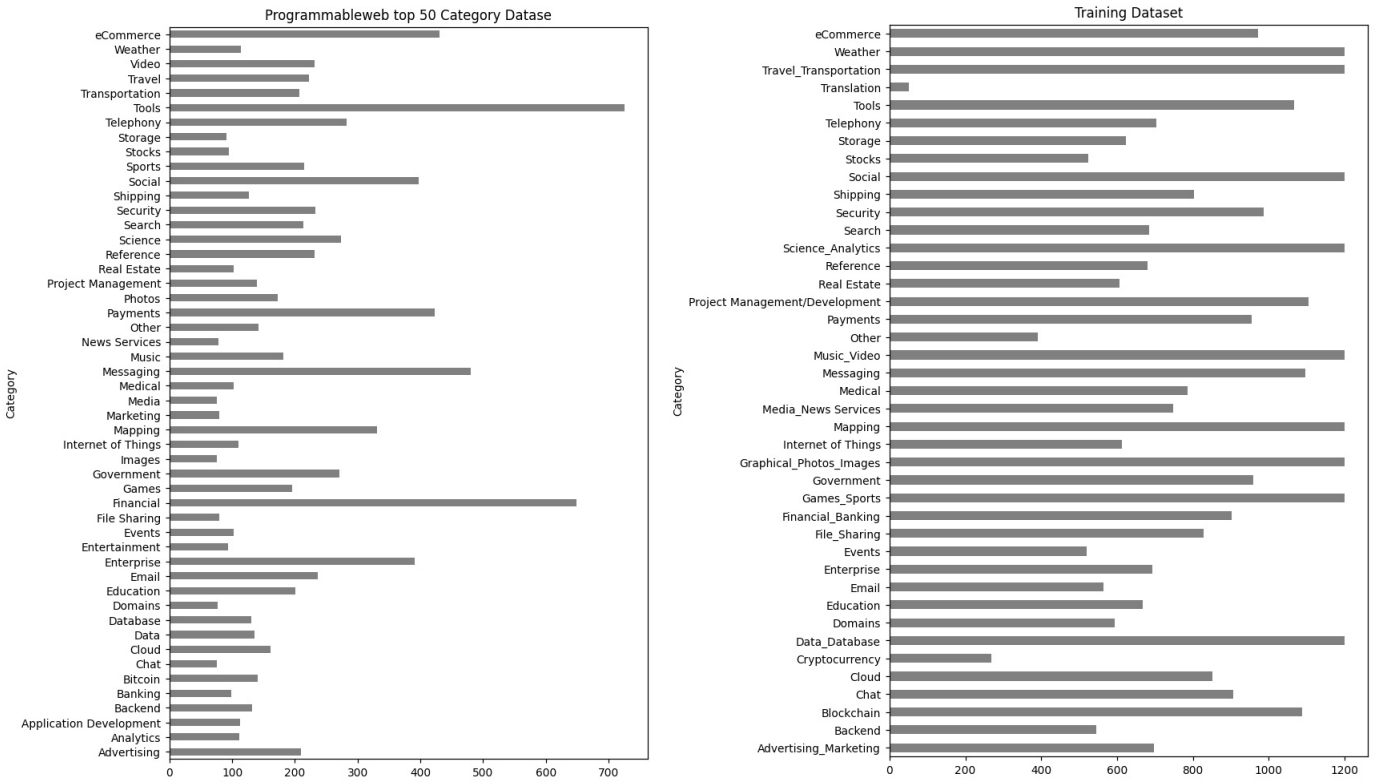

Fig. 3. ProgrammableWeb benchmark dataset and final training dataset data skew

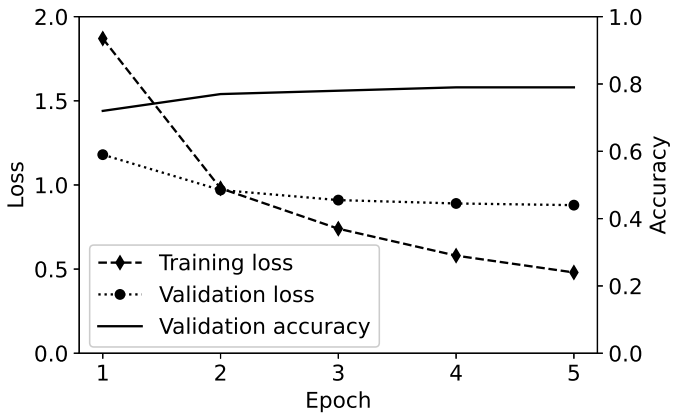

Fig. 4. Hyperparameters tunning process and results

proposed approach to classify web services unrelated to the training set. The web services of the testing set are obtained from industry, whereas the training set comprises of OSS services. The empirical study was designed based on the guidelines of Runeson et al. [28] and is reported according to the Linear Analytic Structure.

A. Objective and Research Questions

The goal of our study, formulated through the Goal-Question-Metric approach is: to ***evaluate*** the SMART-CLIDE service classification approach ***with respect to*** its performance, ***from the point of view of*** accurate classification, ***in the context of*** service-based software development. Based on this goal, this study examines the following research questions.

RQ_1 What is the accuracy of the service classification approach?

This question aims at answering the primordial question of how well the approach performs under a non-trivial, industrial scenario. Considering the dimension of the solution space (i.e., the number of categories) and the context of the study (i.e., a model trained with OSS services and tested on industrial services), we seek to explore a variety of indicators to better characterize the model's performance. For instance, we explore the behaviour of the algorithm in cases when expert judgement is solid, compared to others where it is more blurry.

RQ_2 Are there patterns for classification success or failure?

On top of understanding the model's performance numerically, we also seek potential, underlying rationale to incorrect classifications. To this end, we explore sets of specific service categories that are usually obscure and can cause classification problems. The answer to this question supports a more appropriate employment of the model, informing users about the reliability of the predictions; as

well as, provides information that can support the development of future research endeavours.

B. Case Description

To evaluate our approach, we designed a multiple case study within two software development companies. The first company is located in Germany, whereas the other in Portugal and are both actively involved in service-based software development. The Portuguese company works on the development of Internet-of-Things (IoT) applications, as well as the provision of a platform that assists their customers to develop their own IoT products, or to use existing services. The German company works on the provision of cloud solutions to their customers.

The units of analysis for this study comprise a set of web services that the companies have developed for their internal purposes and they have stored in SMARTCLIDE's repository. Out of the more than 200 (both OSS and internal) services that they have stored, we have filtered 67 (as units of analysis) that are domain-specific and internal. Thus, we excluded services aimed at providing support for configuring IoT devices or cloud infrastructure, since such services can be used in any application domain operating on top of IoT or cloud. An OSS example of such a cases is the `Arduino core for ESP826`[2], which brings support for the ESP8266 chip to the Arduino environment. `Arduino core for ESP826` comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, and work with SD cards, servos, SPI and I2C peripherals. The final test set of services consist a representative sample, as they address a variety of types of services (see Section V) and are completely orthogonal to the training set of services.

C. Data Collection Process and Study Variables

To answer the questions mentioned in Section IV-A, we contrasted the output of the SMARTCLIDE service classification approach to the expert opinion of engineers of the companies on the application domain that each service falls into. As annotators for the classification of services, we have used four engineers from the Portuguese company, and seven engineers from the German company. All of them are experts in service-based software development and have used at least ones the services that are selected as units of analysis.

To obtain their expert classification, we asked them to fill out an online form where each question provided the service title (1) and description (2) (i.e., the input to the ML model), and asked for a single correspondent category (3) and the confidence (4) on that answer through a five-point Likert scale (from '*Low Confidence*' to '*High Confidence*'). Fig. 5 depicts a sample of the questionnaire, demonstrated on an OSS service that is not part of the actual dataset.

The list of possible categories to choose from was the same ones used as tentative outcomes of the ML model.

[2]https://github.com/esp8266/Arduino

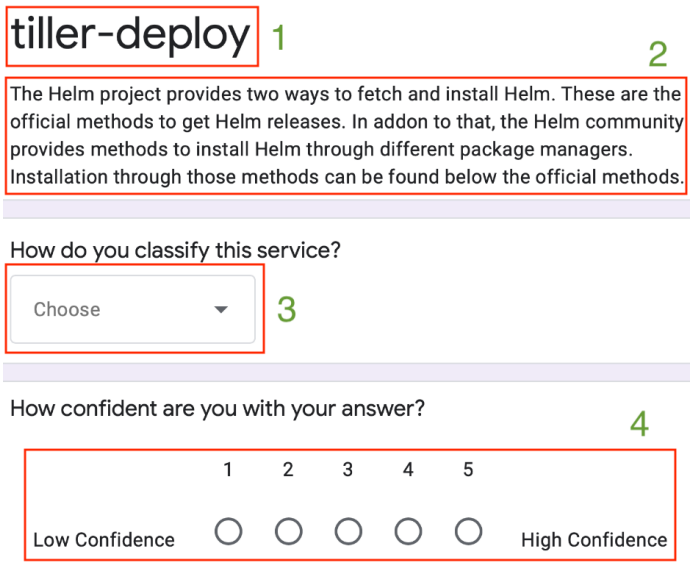

Fig. 5. Questionnaire Sample Question

We note that we did not provide an explanation of the categories to avoid bias. This decision was also driven by the fact that the annotators are experts; thus, we chose to let each one of them to deal with any potential ambiguities or confusion. On the other hand, this choice also provides the annotators with a similar scenario to the ML model training and prediction. Such a design may allow us to better explain model miss-classifications, e.g., if the approach mimics human miss-classifications. At the end of this process, for each unit of analysis, we recorded the triplet `<service information; SMARTCLIDE classification; expert classification>`, where:

- **Service information** refers to a set variables used as input to the proposed approach. In particular, we collect the title and description of each service. These variables were extracted from the description of the git repositories maintained for each service provided by the companies.
- **SMARTCLIDE classification** refers to the output of the SMARTCLIDE service classification model. At this point we need to note that we have excluded the categories 'Internet of Things' and 'Cloud', as services that would fall only into these categories have not been considered as units of analysis (see Section 4.2).
- **Expert classification** refers to the application domain in which a service falls in. We note that in case of disagreement among the expert evaluations, a service might be assigned to more than one application domain. This variable is recorded after applying the following procedure: (1) collect the application domains for the unit of analysis from all expert annotators; (2a) in case of 75% agreement (or more), assign the prevalent application domain to the variable; (2b) in cases of an agreement below 75%, discuss to resolve conflicts, and compile a list application domains to this variable.

We note that since all data of this study are proprietary, we are not able to provide a full replication package. For the sake of replication, the interested stakeholders can use the pre-trained ML models that are provided in SMARTCLIDE's repository.

D. Data Analysis

To answer **RQ$_1$**, we assess the correctness of the model, by calculating the *accuracy* metric. To explore if the model produces more accurate classification for units of analysis for which the experts were more confident, we calculated accuracy for the following ranges of experts' confidence[3] '[1.0–2.5]', '(2.5–3.0]', '(3.0–3.5]', '(3.5–4.0]', and '(4.0–5.0]'. We expect that as the confidence of the experts increases, so will the accuracy of the model. For cases with low confidence from experts, the model is expected to face ambiguity problems. Finally, to investigate the same issue, we perform hypothesis testing to explore if units that are classified correctly are on average associated with higher experts' confidence (compared to miss-classified services). To answer **RQ$_2$**, we revisited the results of RQ$_1$with the particular purpose of identifying patterns (i.e., specific sets of application domains) of incorrect classifications. To achieve this goal, we performed a qualitative analysis on the findings of step (2b) of data collection–see Section IV-C.

V. Results and Interpretation

In this section, we present the results of our analysis, organized by research question: Section V-A deals with evaluating the accuracy of the proposed approach; whereas Section V-B with the assessment of the service categories. In this section, along with presenting the raw results, we attempt an initial interpretation, also comparing with other similar studies from the literature.

A. RQ$_1$: Accuracy of SMARTCLIDE service classification approach

In this section, we answer RQ$_1$ i.e., evaluate the correctness of the model, also considering the confidence of expert evaluations. Based on our analysis, the accuracy of the model using the complete dataset (i.e., regardless of the confidence of the expert annotators) is 77.61% (52 correct and 15 incorrect classifications). The achieved accuracy is rated as very satisfactory, since compared to the literature: (a) this study uses a substantially larger set of application domains—constituting the classification far more challenging; and (b) the study setup aimed at generalization to different contexts, i.e., testing an OSS-based trained model on industrial data. To assess the ability of the proposed approach to work on training and testing sets with different characteristics, we have examined the service description of both training set (with OSS services) and testing set (with industrial services). The results on the size of the description are presented in Fig. 6. The length of service description in the industrial dataset is shorter in average. For most of the services, the range of length

[3]Low Confidence = 1.0; High Confidence = 5.0

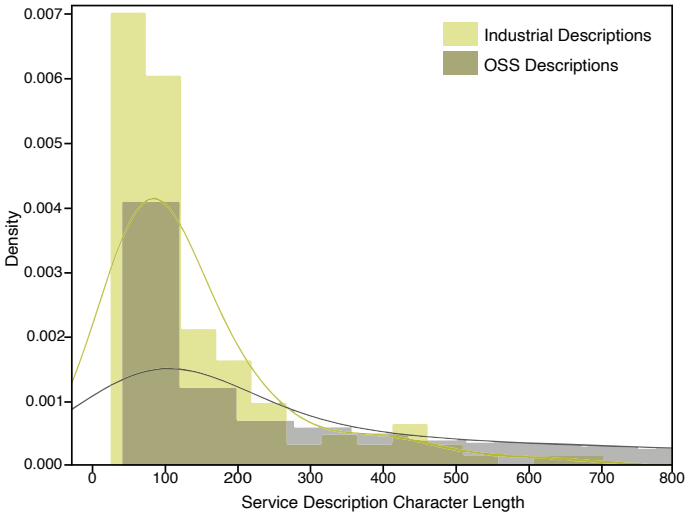

Fig. 6. Length of service descriptions in training and testing sets

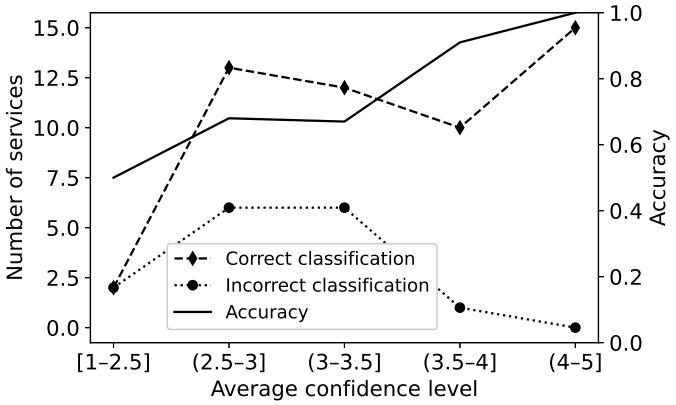

Fig. 7. Approach Performance vs. Experts Confidence

is less than 400 characters, whereas for the training set the majority of services had almost an one-paragraph description of service functionality. However, this difference in description does not seem to affect the accuracy of the proposed approach.

Next, we explore the relation between model accuracy and the confidence of experts' judgements. The results are summarized in Fig. 7. Based on the findings (continuous line), we can observe that the accuracy of the model is increasing. On the one hand, for units of analysis when the annotators declared an averagely high confidence (i.e., '(4–5]'), the accuracy is 100% (15 services classified correctly). On the other hand, for the four services that the experts declared an averagely low confidence (i.e., '[1.0–2.5]'), the model showcased an accuracy of 50%. This finding is considered intuitive, since ML mechanisms would likely learn the human expert fuzziness, resulting in uncertain categorization.

To explore if the aforementioned observation (i.e., incorrect classifications are concentrated around low confidence values, and vice-versa), we performed an independent sample t-test. The `t-value` of the test is 3.408 (`p-value` $<$ 0.01),

suggesting that *the difference between the mean confidence values for the sets of correct and incorrect classifications of the model is statistically significant*.

B. RQ_2: Model Qualitative Assessment

In this section, we dig deeper into insights that our analysis has offered, contributing to understanding the underlying relations between service categories (as retrieved from the literature—see Section III). We start by looking into the first round of merging which was performed at the step 2 of the proposed ML approach (***Dealing with class diversity using service category clustering***). That step led to 6 merges, which were reviewed by the researchers and have been considered as intuitive. Therefore, these categories were merged before running the model on the test data, and were presented as merged to experts. Despite the first round of merging, during the data collection (see **Expert classification** in Section IV-C), additional candidate merges of categories have been identified. While discussing cases of low agreement levels among experts, the experts identified two sources of confusion: (a) "*the provided classes are quite generic, and they are not at the same level of granularity*"; and (b) "*a service might indeed fit two domains, e.g., a sensor for temperature can be used for Sports apps, as well as Weather apps*".

The list of merged categories is presented in Table I (light grey: first merging round, and dark grey lines from the second merging round). After merging the proposed sets of application domains, the classification model that we propose contains 12 categories with combined domains, and 26 categories with a single domain (not presented in Table I). We note that revised sets of categories (group of two in light grey, transformed to a group of three in dark grey) are only counted once. All the merges can be considered intuitive and meaningful, since:

- ***Database*** services deal with ***Data***; and ***Analytics*** services run on ***Data***;
- a large portion of ***Security*** services run on checking ***emails*** for spams;
- ***Media*** (apart from communication) can be perceived as ***Storage*** media;
- ***Medical*** services are always ***Scientific***, and often rely on ***Data Analytics***;
- ***Messaging*** services are triggered by ***Events***;
- ***Photos*** and ***Images*** and ***Project Development*** and ***Management*** are very closely linked conceptually;
- ***Scientific*** services usually rely on data ***Analytics***;
- ***Social*** media include instant ***Messaging*** services;
- ***Advertisement*** is a special task of ***Marketing***;
- ***Travel*** and ***Transportation*** are conceptually related, often using ***Maps***;
- ***Weather*** applications are always ***Scientific***.

TABLE I
MERGING OF SERVICE CATEGORIES

Data with Database
Data with Database with Analytics
Email with Security
Media with Storage
Medical with Science with Analytics
Messaging with Events
Photo with Images
Project Development with Project Management
Science with Analytics
Social with Messaging
Advertizing with Marketing
Travel with Transportation
Travel with Transportation with Mapping
Weather with Science

VI. IMPLICATIONS TO RESEARCHERS AND PRACTITIONERS

In this section we discuss the main implications of this work to researchers and practitioners. On the one hand, we encourage ***practitioners*** to use the proposed approach through the already existing SMARTCLIDE front-end or API, since it seems able to provide satisfactory classification of services, even for partially unrelated data. Additionally, regarding service specification, we encourage practitioners to enlarge the description of services, so that their purpose becomes more explicit to other stakeholders, as well as automatic classifiers. Finally, we prompt practitioners to consider the similarities between specific application domains (as presented in Table. I), when performing domain-specific reuse, i.e., if a service is not identified using the primary application domain (e.g., 'Weather') we propose to search for services in related domains (e.g., 'Science' or 'Data Analysis').

On the other hand, regarding ***researchers***, we encourage the adoption of the strategy defined in this study to boost the generalization of the obtained results. In particular, when possible, we propose to go beyond cross-project validation, and test models trained on OSS data, with industrial data. The results of our study suggest that achieving a satisfactory level of accuracy in classifying industrial services is possible when training a model with a plethora of OSS services. This finding is also supported by the literature, suggesting that closed- and "*well-established*" open-source projects are not "*very different*" in terms of internal structure, functionality, and documentation. Additionally, we encourage researchers that are interested in service classification to consider the tentative overlap of service categories identified by the experts of this study (presented in Table. I), as they can help both in the design and evaluation of similar classification approaches.

VII. THREATS TO VALIDITY

This work, as any other kind of empirical study in software engineering, suffers from various threats to validity. *Internal validity* is not present to this study, since we do not aim at drawing causal relations. Regarding *external validity*, despite

our intention to decouple the training and the test set, both in terms of overlap and context, we cannot guarantee the same accuracy in datasets of other companies. Nevertheless, the results are promising in this direction. With respect to *reliability*, the study design is described in detail in this paper, and the pre-trained ML models and APIs for invoking the classifiers are provided in SMARTCLIDE's repository. Thus, reproducibility is supported. However, due to the NDAs that apply within SMARTCLIDE, we are not able to disclose the complete dataset; thus, impacting replicability. Finally, we have identified threats to *construct validity*. The results of our study are related to the selected service classification schema, and a different schema may yield different results. However, the selected schema is well-established in the domain of service classification, and it has been validated by both the authors and the industrial experts.

VIII. Conclusions

In this paper we focused on domain-specific service reuse, and proposed a ML approach that is able to classify web services, solely based on their textual description to approx. 40 pre-existing service categories. The ML model has been trained on thousands of OSS service data, and has been tested on 67 industrial, closed-source services. The results of the validation have been very positive, achieving a perfect classification for services whose human classification was confident. Additionally, the study has unveiled relations among service categories, which potentially obscure the classification process, proposing beneficial category mergings. The results have been interpreted, analysed, and various implications for researchers and practitioners have been identified.

References

[1] J. Wang, J. Yu, P. Falcarin, Y. Han, and M. Morisio, "An approach to domain-specific reuse in service-oriented environments," in *High Confidence Software Reuse in Large Systems*, H. Mei, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 221–232.

[2] C. W. Krueger, "Software reuse," *ACM Computing Surveys (CSUR)*, vol. 24, no. 2, pp. 131–183, 1992.

[3] M. T. Baldassarre, A. Bianchi, D. Caivano, and G. Visaggio, "An industrial case study on reuse oriented development," in *21st IEEE International Conference on Software Maintenance (ICSM'05)*. IEEE, 2005, pp. 283–292.

[4] S. Brinkkemper, S. Jansen, C. Demir, and I. Hunink, "Pragmatic and opportunistic reuse in innovative start-up companies," *IEEE Software*, vol. 25, no. 06, pp. 42–49, 2008.

[5] G. Digkas, N. Nikolaidis, A. Ampatzoglou, and A. Chatzigeorgiou, "Reusing code from stackoverflow: The effect on technical debt," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 87–91.

[6] A. Capiluppi, C. Boldyreff, and K.-J. Stol, "Successful reuse of software components: A report from the open source perspective," in *Open Source Systems: Grounding Research*, S. A. Hissam, B. Russo, M. G. de Mendonça Neto, and F. Kon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 159–176.

[7] M.-E. Paschali, A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, and I. Stamelos, "Reusability of open source software across domains: A case study," *Journal of Systems and Software*, vol. 134, pp. 211–227, 2017.

[8] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service," *Computer*, vol. 36, no. 10, pp. 38–44, 2003.

[9] X. Xu, G. Motta, Z. Tu, H. Xu, Z. Wang, and X. Wang, "A new paradigm of software service engineering in big data and big service era," *Computing*, vol. 100, pp. 353–368, 2018.

[10] H. Yang, *Software reuse in the emerging cloud computing era*. IGI Global, 2012.

[11] C. F. Snook, M. J. Butler, A. Edmunds, and I. Johnson, "Rigorous development of reusable, domain-specific components, for complex applications," in *2014 International Workshop on Critical Systems Development with UML (CSDUML)*, Lisbon, Portugal, 2004, pp. 115–129.

[12] A. Lampropoulos, A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, and I. Stamelos, "React - a process for improving open-source software reuse," in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 2018, pp. 251–254.

[13] S. K. S., A. Ahmed, and M. Shankar, "A composite classification model for web services based on semantic amp; syntactic information integration," in *2015 IEEE International Advance Computing Conference (IACC)*, 2015, pp. 1169–1173.

[14] S. S. Kamath and V. S. Ananthanarayana, "Semantics-based web service classification using morphological analysis and ensemble learning techniques," *International Journal of Data Science and Analytics*, vol. 2, no. 1-2, pp. 61–74, 10 2016.

[15] M. P. Crasso, A. Zunino, and M. R. Campo, "Awsc: An approach to web service classification based on machine learning techniques," *Inteligencia Artif.*, vol. 12, pp. 25–36, 2008.

[16] M. S. Alshafaey, A. I. Saleh, and M. F. Alrahamawy, "A new cloud-based classification methodology (cbcm) for efficient semantic web service discovery," *Clust. Comput.*, vol. 24, pp. 2269–2292, 2021.

[17] W. Li, C. Yang, D. Nebert, R. Raskin, P. Houser, H. Wu, and Z. Li, "Semantic-based web service discovery and chaining for building an arctic spatial data infrastructure," *Computers & Geosciences*, vol. 37, no. 11, pp. 1752–1762, 2011.

[18] R. Nisa and U. Qamar, "A text mining based approach for web service classification," *Information Systems and e-Business Management*, vol. 13, no. 4, pp. 751–768, 2015.

[19] G. Pirro, P. Trunfio, D. Talia, P. Missier, and C. Goble, "Ergot: A semantic-based system for service discovery in distributed infrastructures," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (ICCCGC)*. IEEE, 2010, pp. 263–272.

[20] K. Punitha, "A novel mixed wide and pso-bi-lstm-cnn model for the effective web services classification," *Webology*, vol. 17, no. 2, pp. 218–237, 2020.

[21] M. M. Rahman, C. K. Roy, and D. Lo, "Automatic query reformulation for code search using crowdsourced knowledge," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1869–1924, 2019.

[22] C. Sanchez Sanchez, E. Villatoro Tello, A. G. Ramirez De La Rosa, H. Jimenez Salazar, and D. E. Pinto AvendañO, "WSDL information selection for improving web service classification," *Research in Computing Science*, vol. 144, pp. 83–96, 2017. [Online]. Available: http://ilitia.cua.uam.mx:8080/jspui/handle/123456789/503

[23] Y. Yang, N. Qamar, P. Liu, K. Grolinger, W. Wang, Z. Li, and Z. Liao, "Servenet: A deep neural network for web services classification," in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020, pp. 168–175.

[24] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543.

[25] Z. Alizadeh-Sani, P. Martínez, G. González, A. González-Briones, P. Chamoso, and J. Corchado, "A hybrid supervised/unsupervised machine learning approach to classify web services," in *2021 International Workshops of Practical Applications of Agents and Multi-Agent Systems (PAAMS)*. Cham: Springer, 2021, pp. 93–103.

[26] K. Church, "Word2vec," *Natural Language Engineering*, vol. 23, pp. 155–162, 2017.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, MN, USA: Association for Computational Linguistics, 2019, pp. 4171–4186.

[28] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley Blackwell, 2012.